Fundamentele Computing-ului și nevoia de scalare

Videoclipul începe prin a descrie componentele de bază ale unui computer: hardware-ul (CPU, RAM, disc) și software-ul (sistemul de operare și kernel-ul). Sistemul de operare acționează ca o interfață între software și hardware-ul "bare metal", permițând aplicațiilor să utilizeze resursele fizice ale mașinii.

Se explică apoi modelul **client-server**, în care aplicația ta (clientul) interacționează cu servere aflate la distanță pentru a accesa date sau servicii. Atunci când o aplicație devine populară, traficul crește, necesitând scalarea serverelor. Videoclipul detaliază două metode de scalare:

- **Scalare verticală**: Mărirea resurselor unui singur server (mai mult RAM, un CPU mai puternic). Această metodă are o limită fizică și devine costisitoare.
- **Scalare orizontală**: Distribuirea aplicației pe mai multe servere mai mici, de obicei sub forma de **microservicii**. Acest lucru permite o flexibilitate și o eficiență a costurilor superioare.

Containere vs. Mașini Virtuale

O parte esențială a videoclipului este comparația dintre **containere** și **mașinile virtuale (VM)**.

- **Mașinile virtuale** folosesc un hypervisor pentru a rula mai multe sisteme de operare izolate pe aceeași mașină fizică. Fiecare VM include propriul kernel și propriile biblioteci, fiind o soluție ce consumă multe resurse.
- **Containerele Docker** oferă o abordare mai eficientă. Ele utilizează kernel-ul sistemului de operare gazdă și izolează aplicațiile la un nivel superior, doar la nivel de procese. Astfel, fiecare container este mult mai ușor, se pornește rapid și consumă mai puține resurse. Aceasta se numește **virtualizare la nivel de sistem de operare**.

Cum funcționează Docker: Fișier, Imagine, Container

Videoclipul explică ciclul de viață al unei aplicații containerizate în trei etape clare:

1. **Fișierul Docker (Dockerfile)**: Este un document text care conține toate instrucțiunile necesare pentru a construi un mediu de rulare pentru aplicația ta. Acesta definește sistemul de operare de bază, dependințele, variabilele de mediu, porturile și comanda de start a aplicației.
2. **Imaginea Docker (Image)**: Este un șablon static, imutabil, creat din fișierul Docker. O imagine include codul aplicației, bibliotecile, fișierele sistemului de operare și configurația. Poate fi stocată într-un registru (precum **Docker Hub**) și distribuită, asigurând că aplicația rulează la fel oriunde.
3. **Containerul Docker (Container)**: Este o instanță rulabilă, izolată, a unei imagini. Un container este efemer și portabil, putând fi pornit și oprit cu ușurință.

Elemente practice și comenzi Docker

Videoclipul oferă și o demonstrație practică, prezentând o serie de comenzi esențiale:

- `FROM`: Specifică imaginea de bază.
- `WORKDIR`: Setează directorul de lucru.
- `COPY`: Copiază fișierele din mașina gazdă în container.
- `RUN`: Rulează comenzi în timpul construcției imaginii.
- `EXPOSE`: Specifică portul pe care va rula aplicația.
- `CMD`: Definește comanda care se execută la pornirea containerului.

Sunt menționate și comenzi din linia de comandă (CLI) pentru gestionarea containerelor, cum ar fi `docker build`, `docker run`, `docker stop`, `docker push` și `docker pull`. De asemenea, este prezentat pe scurt **Docker Desktop** ca o interfață grafică pentru a interacționa cu Docker.

Concepte avansate: Docker Compose și Kubernetes

Pentru a ilustra cum se gestionează aplicațiile mai complexe, videoclipul introduce două concepte avansate:

- **Docker Compose**: Un instrument pentru a defini și a rula aplicații cu mai multe containere (ex: un backend, un frontend și o bază de date) folosind un singur fișier YAML.
- **Kubernetes**: Un sistem de orchestrare open-source pentru automatizarea implementării, scalării și gestionării aplicațiilor containerizate. Videoclipul menționează că Kubernetes este util pentru aplicații la scară mare și oferă concepte precum "nodes" (mașini) și "pods" (containere grupate) pentru a atinge o toleranță înaltă la erori.